生物系の数理科学 第15回

山口 諒

パーセプトロン・ニューラルネットワーク

1. はじめに

人工ニューラルネットワーク（Artificial Neural Network: ANN）のルーツは、脳の神経細胞（ニューロン）がシナプスを通して信号をやり取りする仕組みを数理モデル化しようとした試みにある。その最も初期の形の一つが**パーセプトロン**（Perceptron）であり、1950 年代後半にフランク・ローゼンブラット（Frank Rosenblatt）によって提唱された。以後、パーセプトロンは多層化や活性化関数の多様化を経て、現代のニューラルネットワーク（深層学習）へと発展を遂げている。ここでは、まずパーセプトロンの数式的枠組みを整理し、そこからニューラルネットワークへ拡張されていった歴史や神経科学との関連を概説する。

2. 人工ニューロン

2.1 総入力と決定関数

人工ニューロンの入力として、ベクトル $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ を考える（m 次元）。それぞれは特徴量などの値である。入力それぞれについて重みベクトル $\boldsymbol{w} = (w_1, w_2, \ldots, w_m)$ を考え、ニューロンの内部で「重み付き和」を計算する。さらにバイアス b を加えることで総入力（net input）z を得る：

$$z = \boldsymbol{w}^\top \boldsymbol{x} + b.$$

あるいは、成分表示で書くと

$$z = w_1\, x_1 + w_2\, x_2 + \cdots + w_m\, x_m + b.$$

この z が**決定関数**（decision function）の入力に相当し、ニューロン出力をどう返すかが次に決まる。

2.2 ステップ関数 $\sigma(z)$

単純パーセプトロンでは、出力を 2 値 $\{0, 1\}$ または $\{-1, +1\}$ で返す。しばしば、

$$\sigma(z) = \begin{cases} 1, & (z \geq 0), \\ 0, & (z < 0), \end{cases}$$

という**単位ステップ関数 (unit step function)** を使う。あるいは

$$\sigma(z) = \begin{cases} +1, & (z \geq 0), \\ -1, & (z < 0), \end{cases}$$

という関数の場合もある。

2.3 二値分類への応用

最終的に、パーセプトロンの**決定境界**は $z = 0 \iff \boldsymbol{w}^\top \boldsymbol{x} + b = 0$ という超平面である。2 次元 (x_1, x_2) なら直線で分割する。

$$y = \begin{cases} 1, & (\boldsymbol{w}^\top \boldsymbol{x} + b \geq 0), \\ 0, & \text{otherwise}. \end{cases}$$

あるいは $y \in \{-1, +1\}$ の場合でも同じ原理で、$z = 0$ が境界となる。

3. パーセプトロンの学習規則

3.1. はじめに

ここまで、**パーセプトロン**（あるいは MCP ニューロン）自体の構造——つまり

$$z = \boldsymbol{w}^\top \boldsymbol{x} + b, \quad \text{output} = \begin{cases} 1 & (z \geq 0), \\ 0 & (z < 0), \end{cases}$$

を示した。ここからは、「どうやって重み $\boldsymbol{w}$ やバイアス b を学習するか」が問題になる。フランク・ローゼンブラット (1957) によるパーセプトロンの初期学習則は「誤分類が起きたときにだけ重みを更新する」という**簡単な手順**であり、以下のステップで進行する。

3.2. パーセプトロンの学習規則の手順

3.2.1 アルゴリズムの流れ

1. **重みとバイアス**を初期化：0、もしくは小さい乱数などで $w_j(0), b(0)$ を設定。
2. **訓練データ** $(\boldsymbol{x}^{(i)}, t^{(i)})$ を 1 つずつ取り出して、以下を繰り返す:
   1. (出力値の計算) $z^{(i)} = \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b$. 出力 $\text{output}^{(i)} = \sigma\big(z^{(i)}\big)$ (ステップ関数)。
   2. (重みとバイアスの更新)

$$w_j \leftarrow w_j + \Delta w_j, \quad b \leftarrow b + \Delta b.$$

ここで Δw_j と Δb は以下の学習則で定義される（誤差があるときだけ更新）。

3.3. 具体的な更新則

$$\Delta w_j = \eta \left(t^{(i)} - \text{output}^{(i)} \right) x_j^{(i)},$$
$$\Delta b = \eta \left(t^{(i)} - \text{output}^{(i)} \right).$$

ここで

- $t^{(i)} \in \{0, 1\}$ あるいは $\{-1, +1\}$：訓練サンプル i の正解ラベル
- $\text{output}^{(i)}$：パーセプトロンが計算した 0/1 (または -1/+1) の予測
- $\eta > 0$：学習率 (learning rate)

誤差 $(t^{(i)} - \text{output}^{(i)})$ が 0 であれば更新なし。予測が正解の場合は重みを変えない。予測が間違いなら、正例を負例と判定した場合には $t^{(i)} - \text{output}^{(i)} = 1 - 0 = 1$ (二値設定)、それに $\eta x_j^{(i)}$ を足し重みを増やす。逆に負例を正例と判定したなら $1 - 0$ ではなく 0-1=-1、など適宜符号が変わって、重みがその方向に調整される。これを訓練データに対して何周も繰り返す (エポック)。

理論上、もし訓練データが線形分離可能であれば、このアルゴリズムは有限ステップで誤分類 0 の重みベクトルを見つけられる（**パーセプトロン収束定理**）。ただし、データが線形分離不可能なら学習は発散するか、無限ループになる可能性がある。

4. ADALINE の学習規則と勾配降下法

4.1. はじめに

パーセプトロン学習則は「誤分類が起きたサンプルに対してのみ重みを更新する」方式だった。一方、**ADALINE** (Adaptive Linear Neuron) や **Widrow–Hoff 学習**では、**連続値の出力**を想定し、二乗誤差 (MSE) を最小化する方針をとる。これにより、**勾配降下法 (gradient descent)** を用いたなめらかな学習が可能となる。

4.2. ADALINE の損失関数：MSE (平均二乗誤差)

4.2.1 連続出力の定義

ADALINE では、ニューロン出力がパーセプトロンのような $\{0, 1\}$ (または $\{-1, +1\}$) ではなく、**連続値** $\hat{y}^{(i)} = z^{(i)}$ として扱う。具体的には、総入力

$$z^{(i)} = \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b$$

を、そのまま出力値 (net output) とする（ステップ関数を通さない）。

4.2.2 目的関数 = 損失関数 $L(w, b)$

各サンプル $i = 1, \ldots, n$ について、正解ラベル $y^{(i)}$ (連続 or 2 値ラベルを取りうるが、ここでは実数として扱う) と予測 $\hat{y}^{(i)} = z^{(i)}$ の二乗誤差を考え、「平均二乗誤差 (MSE)」を以下のように定義する:

$$L(\boldsymbol{w}, b) = \frac{1}{2n} \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)} \right)^2.$$

($\frac{1}{2n}$ の $\frac{1}{2}$ は微分の際に都合がよいために付ける。)

4.3. 勾配降下法：$\nabla L(\boldsymbol{w}, b)$ を用いたパラメータ更新

4.3.1 パラメータ更新の基本形

ADALINE では、重み $\boldsymbol{w}$ とバイアス b を、損失関数 L を最小化する方向に微少移動させる。すなわち、勾配降下 (gradient descent) と呼ばれる以下の規則にしたがって計算を進める:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial L}{\partial \boldsymbol{w}}, \quad b \leftarrow b - \eta \frac{\partial L}{\partial b}.$$

ここ $\eta > 0$ は学習率 (learning rate) である。式を整理して、

$$\Delta \boldsymbol{w} = -\eta \nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b), \ \Delta b = -\eta \frac{\partial L}{\partial b}.$$

最終的に

$$\boldsymbol{w} := \boldsymbol{w} + \Delta \boldsymbol{w}, \quad b := b + \Delta b.$$

4.3.2 勾配 $\frac{\partial L}{\partial w_j}$ の計算

続いて、損失関数

$$L(\boldsymbol{w}, b) = \frac{1}{2n} \sum_{i=1}^{n} \Big(y^{(i)} - z^{(i)} \Big)^2$$

(ここ $z^{(i)} = \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b$) の偏微分を求める。まず w_j 成分について。

$$\frac{\partial L}{\partial w_j} = \frac{1}{2n} \sum_{i=1}^{n} \frac{\partial}{\partial w_j} \Big(y^{(i)} - z^{(i)} \Big)^2.$$

チェーンルール:

$$\frac{\partial}{\partial w_j} \Big(y^{(i)} - z^{(i)} \Big)^2 = 2 \left(y^{(i)} - z^{(i)} \right) \frac{\partial}{\partial w_j} \Big(y^{(i)} - z^{(i)} \Big).$$

ただし $y^{(i)}$ は定数 (正解ラベル) なので微分は 0、$z^{(i)} = \sum_k w_k x_k^{(i)} + b$, したがって

$$\frac{\partial}{\partial w_j}(-z^{(i)}) = -\frac{\partial}{\partial w_j} \Big(\sum_k w_k x_k^{(i)} + b \Big) = -x_j^{(i)}.$$

よって

$$\frac{\partial}{\partial w_j} \Big(y^{(i)} - z^{(i)} \Big)^2 = 2 \left(y^{(i)} - z^{(i)} \right) \cdot (-x_j^{(i)}).$$

$$= -2\, x_j^{(i)} \left(y^{(i)} - z^{(i)} \right).$$

故に

$$\frac{\partial L}{\partial w_j} = \frac{1}{2n} \sum_{i=1}^{n} \Big[-2\, x_j^{(i)} \left(y^{(i)} - z^{(i)} \right) \Big].$$

$$= -\frac{1}{n} \sum_{i=1}^{n} x_j^{(i)} \big(y^{(i)} - z^{(i)} \big).$$

マイナスを外に出すと

$$\frac{\partial L}{\partial w_j} = -\frac{1}{n}\sum_{i=1}^{n}\big(y^{(i)} - z^{(i)}\big)\, x_j^{(i)}.$$

4.3.3 バイアス b への偏微分

同様に $z^{(i)} = \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b$, なので

$$\frac{\partial z^{(i)}}{\partial b} = 1.$$

したがって

$$\frac{\partial L}{\partial b} = \frac{1}{2n}\sum_{i=1}^{n}\frac{\partial}{\partial b}\Big(y^{(i)} - z^{(i)}\Big)^2 = \frac{1}{2n}\sum_{i=1}^{n} 2\big(y^{(i)} - z^{(i)}\big)\cdot\big(-1\big).$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\big(y^{(i)} - z^{(i)}\big).$$

4.4. 更新式

したがって勾配降下法に基づく重み更新は

$$\Delta w_j = -\,\eta\,\frac{\partial L}{\partial w_j} = -\eta\,\Big[-\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - z^{(i)})\, x_j^{(i)}\Big] = \frac{\eta}{n}\sum_{i=1}^{n}\big(y^{(i)} - z^{(i)}\big)\, x_j^{(i)}.$$

$$\Delta b = -\eta\,\frac{\partial L}{\partial b} = -\eta\big[-\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - z^{(i)})\big] = \frac{\eta}{n}\sum_{i=1}^{n}\big(y^{(i)} - z^{(i)}\big).$$

最終的に:

$$w_j \leftarrow w_j + \Delta w_j, \quad b \leftarrow b + \Delta b,$$

すなわち

$$w_j := w_j + \frac{\eta}{n}\sum_{i=1}^{n}\Big(y^{(i)} - z^{(i)}\Big)\, x_j^{(i)},$$

$$b := b + \frac{\eta}{n}\sum_{i=1}^{n}\Big(y^{(i)} - z^{(i)}\Big).$$

$z^{(i)} = w_1 x_1^{(i)} + \cdots + w_m x_m^{(i)} + b$ は各サンプルの予測値 (= net output) となる。これをすべてのサンプルに対して同時に更新する（フルバッチ勾配降下）こともあれば、ミニバッチで逐次的に行う場合もあるが、原理は同じである。

5. 大規模データの機械学習と確率的勾配降下法：ADALINE 学習の拡張

5.1. はじめに

前回の解説では、**ADALINE** の学習規則をフルバッチ勾配降下法（batch gradient descent）で示した。すなわち、全訓練データに対して損失関数 $L(\boldsymbol{w}, b)$ を計算し、その勾

配をまとめて求めて**一括更新**する方法である。しかし、データが数万〜数千万件に及ぶ場合、1 ステップ更新ごとに全データを走査するのは計算コストが膨大となる。そこで実用上は、

確率的 (確率論的) 勾配降下法 (SGD),　ミニバッチ勾配降下法

などの手法が使われる。これらはいずれも**勾配降下**の変形であり、多くの機械学習ライブラリが標準実装している。

5.2. 確率的勾配降下法 (Stochastic Gradient Descent; SGD)

5.2.1 発想

フルバッチの代わりに、「訓練データ 1 件につき 1 回ずつ更新」する方法を**確率的勾配降下 (Stochastic Gradient Descent)** と呼ぶ。つまり以下のように進める：

1. シャッフルされた訓練サンプル列 $(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots$ を用意。
2. 各サンプルが来るたびに、**1 サンプル分の損失** $L_i(\boldsymbol{w}, b) = \frac{1}{2}\big(y^{(i)} - z^{(i)}\big)^2$ の勾配を計算し、パラメータを更新する：

$$\Delta w_j = \eta \left(y^{(i)} - z^{(i)}\right) x_j^{(i)}, \quad \Delta b = \eta \left(y^{(i)} - z^{(i)}\right).$$

3. これを全データに対して繰り返す（1 エポック）。
4. エポック間にデータ順序を再度シャッフルして、過学習や局所解への執着を防ぐ。

こうすると 1 ステップの計算は非常に軽くなるが、勾配推定の分散が大きい。結果として学習がややノイズを含む軌道を辿るが、大規模データでは速やかに良い解へ近づくことが期待できる。

5.3. ミニバッチ勾配降下法 (Mini-batch)

5.3.1 定義

ミニバッチとはフルバッチ (n 件まとめ) と SGD(1 件ずつ) の中間で、データを小分けにしてバッチサイズ B 件ごとに更新する方法。更新式は、

$$\Delta w_j = \frac{\eta}{B} \sum_{\ell=1}^{B} \left(y^{(\ell)} - z^{(\ell)} \right) x_j^{(\ell)},$$

$$\Delta b = \frac{\eta}{B} \sum_{\ell=1}^{B} \left(y^{(\ell)} - z^{(\ell)}\right),$$

(ℓ はミニバッチ内のサンプルインデックス)。こうして複数件で勾配を平均し、1 回更新する。